# Timed Game Abstraction of Control Systems*

Christoffer Sloth
Department of Computer Science
Aalborg University
9220 Aalborg East, Denmark
csloth@cs.aau.dk

Rafael Wisniewski
Section of Automation & Control
Aalborg University
9220 Aalborg East, Denmark
raf@es.aau.dk



## ABSTRACT

This paper proposes a method for abstracting control systems by timed game automata, and is aimed at obtaining automatic controller synthesis.

The proposed abstraction is based on partitioning the state space of a control system using positive and negative invariant sets, generated by Lyapunov functions. This partitioning ensures that the vector field of the control system is transversal to the facets of the cells, which induces some desirable properties of the abstraction. To allow a rich class of control systems to be abstracted, the update maps of the timed game automaton are extended.

Conditions on the partitioning of the state space and the control are set up to obtain sound abstractions. Finally, an example is provided to demonstrate the method applied to a control problem related to navigation.

## Categories and Subject Descriptors

B.1.2 [**Control Structures and Microprogramming**]: Control Structure Performance Analysis and Design Aids—*Automatic synthesis, Formal models*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata, Relations between models*

## General Terms

Theory

## Keywords

Abstraction, Automatic controller synthesis, Timed game, Lyapunov function

## 1. INTRODUCTION

*This work was supported by MT-LAB, a VKR Centre of Excellence.

Controller design has been studied for many decades in the control community. In these studies, the primary objectives have been asymptotic stability and disturbance attenuation. This type of controller design is quite mature and can be used to synthesize controllers via, e.g., LMI-based method for linear systems. However, for nonlinear systems, design methods are limited and a considerable amount of manual labor is required in the controller design.

Controller design has also been considered in the computer science community for, e.g., discrete event systems and timed game automata. The requirements for such a system are primarily based on reachability of the system and temporal properties of the system. Especially, the timing requirements used in computer science are very different from requirements known in control theory, as these are defined for a finite time horizon; whereas, control theory is concerned with convergence, i.e., system properties when time goes to infinity. Fully automated tools have been developed for controller synthesis of discrete event systems and timed game automata. These are based on formal verification methods; therefore, the designed controllers are correct-by-design, i.e., the closed-loop control system is guaranteed to comply with the specification. This, in principle, eliminates the need for simulating the closed-loop control systems to perform further verification.

The goal of this paper is to abstract control systems by an automata-based model, and thereby allow automatic controller synthesis. In this way we are able to specify requirements in terms of Timed Computation Tree Logic (TCTL) specifications [1]; hence, requirements to reachability and timing can be added to the usual stability requirement.

Methods for synthesizing controllers for this kind of specification have been proposed in the computer science community for timed game automata [3]. For games the controller is called a strategy, and it decides among the possible choices in the game. The strategy can be automatically synthesized using tools such as UPPAAL Tiga [4].

Methods from formal verification have already been adopted in control theory for controller design in [9]. Here the controller is synthesized to avoid certain unsafe states. For this purpose, a concept of approximate bisimulation, a relaxation of exact bisimulation, has been introduced. This is further demonstrated in [8], where a robot is controlled to avoid some obstacles using a temporal logic specification. How-

ever, the generation of the models used for the synthesis procedure of these methods is based on simulating the system, which makes the method computationally demanding.

Methods for discretized models also exist, where solutions to the system equations are not utilized. One such method is presented in [10], where the principle of control to facet is utilized to synthesize a control strategy.

In this paper, we abstract control systems by timed game automata with an extended update map, and use ideas similar to the bisimulation functions used in [9] in the abstraction procedure. However, the method does not require solutions to the system equations and is therefore not as computationally expensive. The work is an extension of the abstraction procedure presented in [13], which applies for autonomous systems. This abstraction of dynamical systems by timed automata is based on partitioning the state space using Lyapunov functions. The intersections of sub-level sets of Lyapunov functions are used to form the cells that discretize the state space. This makes the problem of synthesizing a control strategy similar to control to facet, as the cells are generated using intersections of invariant sets. We provide a method for the design of switched controllers. The main result of the paper is Theorem 1, which states sufficient and necessary conditions for sound abstractions by timed game automata. Since we abstract the control systems by timed game automata with a modified update map, the synthesis procedure cannot yet be accomplished using existing tools.

This paper is organized as follows: Section 2 contains preliminary definitions used throughout the paper, Section 3 explains the partitioning of the state space and the control, and Section 4 describes how a timed game is generated from the partition. In Section 5, conditions for soundness are set up, Section 6 provides an example, and Section 7 comprises conclusions.

## 1.1 Notation

The set $\{1,\dots,k\}$ is denoted $\boldsymbol{k}$. $B^A$ is the set of maps $A \to B$, $C(\mathbb{R}^n,\mathbb{R}^m)$ is the set of continuous maps $\mathbb{R}^n \to \mathbb{R}^m$. The power set of $A$ is denoted $2^A$. Given a vector $a \in \mathbb{R}^n$, $a(j)$ denotes the $j^{\text{th}}$ coordinate of $a$. Given a set $A$, the cardinality of the set is denoted $|A|$. Consider the Euclidean space $(\mathbb{R}^n,\langle,\rangle)$, where $\langle,\rangle$ is the scalar product. The state space is a connected subset $X \subset \mathbb{R}^n$ such that there exists an open set $U$ such that $\mathrm{cl}(U) = X$. Whenever $f : X \to \mathbb{R}$ is a function and $a \in \mathbb{R}$, we write $f^{-1}(a)$ to shorten the notation of $f^{-1}(\{a\})$.

# 2. PRELIMINARIES

The purpose of this section is to provide definitions related to control systems and timed game automata.

A control system $\Gamma = (X,U,f)$ has state space $X \subseteq \mathbb{R}^n$, input space $U \subseteq \mathbb{R}^m$, and dynamics described by ordinary differential equations $f : X \times U \to \mathbb{R}^n$

$$\dot{x} = f(x,u). \tag{1}$$

The input $u$, is controlled via a continuous map $g : X \to U$.

The system $\Gamma = (X,U,f)$ with the control $g$ is denoted $\Gamma_g = (X,f_g)$, where

$$\dot{x} = f_g(x) = f(x,g(x)). \tag{2}$$

We assume that $f_g$ is locally Lipschitz and has linear growth, then there exists a unique solution of (2) on $(-\infty,\infty)$ [5].

The solution of (2), from an initial state $x_0 \in X_0 \subseteq X$ at time $t \geq 0$ is described by the flow function $\phi_{\Gamma_g} : [0,\epsilon]\times X \to X$, $\epsilon > 0$ satisfying

$$\frac{d\phi_{\Gamma_g}(t,x_0)}{dt} = f_g\left(\phi_{\Gamma_g}(t,x_0)\right) \tag{3}$$

for all $t \geq 0$.

Lyapunov functions are utilized in stability theory and are defined in the following [12].

Definition 1 (Lyapunov Function). *Let $X$ be an open connected subset of $\mathbb{R}^n$. Suppose $f_g : X \to \mathbb{R}^n$ is continuous and let $\mathrm{Cr}(f_g)$ be the set of critical points of $f_g$. Then a real non-degenerate differentiable function $\varphi : X \to \mathbb{R}$ is said to be a Lyapunov function for $f_g$ if*

$$p \text{ is a critical point of } f_g \Leftrightarrow p \text{ is a critical point of } \varphi$$

$$\dot{\varphi}_g(x) \equiv \sum_{j=1}^{n} \frac{\partial \varphi}{\partial x_j}(x) f_g^j(x) \tag{4a}$$

$$\dot{\varphi}_g(x) = 0 \quad \forall x \in \mathrm{Cr}(f_g) \tag{4b}$$

$$\dot{\varphi}_g(x) \neq 0 \quad \forall x \in X\backslash\mathrm{Cr}(f_g) \tag{4c}$$

*and there exists $\alpha > 0$ and an open neighborhood of each critical point $p \in \mathrm{Cr}(f_g)$, where*

$$||\dot{\varphi}_g(x)|| \geq \alpha ||x-p||^2. \tag{5}$$

Remark 1. *We only require the vector field to be transversal to the level curves of a Lyapunov function $\varphi$, i.e., $\dot{\varphi}_g(x) = \langle \nabla\varphi_g(x), f_g(x)\rangle \neq 0$ for all $x \in X\backslash\mathrm{Cr}(f_g)$, and does not use Lyapunov functions in the usual sense, where the existence of a Lyapunov function implies stability, but uses a more general notion from [12].*

To simplify the notation, we use subscript $g$ on $\dot{\varphi}_g$ to indicate that the control $g$ is applied in the calculation of $\dot{\varphi}_g$.

Definition 2 (Reachable set of Dyn. System). *The reachable set of a dynamical system $\Gamma_g$ from a set of initial states $X_0 \subseteq X$ on the time interval $[t_1,t_2]$ is defined as*

$$\mathrm{Reach}_{[t_1,t_2]}(\Gamma_g,X_0) \equiv \{x \in X | \exists t \in [t_1,t_2],\ \exists x_0 \in X_0 \text{ such that } x = \phi_{\Gamma_g}(t,x_0)\}. \tag{6}$$

The control system will be abstracted by a timed game automaton, which is an extension of a timed automaton [7]. In the definition of a timed automaton, a set of diagonal-free clock constraints $\Psi(C)$ is used for the set $C$ of clocks. $\Psi(C)$ is defined as the set of constraints $\psi$ described by the following grammar:

$$\psi ::= c \bowtie k | \psi_1 \wedge \psi_2, \text{ where } c \in C,\ k \in \mathbb{R}_{\geq 0},\ \text{and } \bowtie \in \{\leq, <, =, >, \geq\}. \tag{7}$$

Note that the clock constraint $k$ should usually be an integer, but in this paper no effort is done to convert the clock constraints into integers. Furthermore, the elements of $\bowtie$ are bold to indicate that they are syntactic operations.

Definition 3 (Timed Automaton). *A timed automaton, $\mathcal{A}$, is a tuple $(E, E_0, C, \Sigma, I, \Delta)$, where*

- *$E$ is a finite set of locations, and $E_0 \subseteq E$ is the set of initial locations.*
- *$C$ is a finite set of clocks.*
- *$\Sigma$ is the set of actions.*
- *$I : E \to \Psi(C)$ assigns invariants to locations.*
- *$\Delta \subseteq E \times \Psi(C) \times \Sigma \times 2^C \times E$ is a finite set of transition relations. The transition relations provide edges between locations as tuples $(e, G_{e\to e'}, \sigma, R_{e\to e'}, e')$, where $e$ is the source location, $e'$ is the destination location, $G_{e\to e'} \in \Psi(C)$ is the guard set, $\sigma$ is an action in $\Sigma$, and $R_{e\to e'} \in 2^C$ is the set of clocks to be reset.*

The semantics of a timed automaton is defined in the following, adopting the notion of [7].

Definition 4 (Clock Valuation). *A clock valuation on a set of clocks $C$ is a mapping $v : C \to \mathbb{R}_{\geq 0}$. The initial valuation $v_0$ is given by $v_0(c) = 0$ for all $c \in C$. For a valuation $v$, $t \in \mathbb{R}_{\geq 0}$, and $R \subseteq C$, the valuations $v + t$ and $v[R]$ are defined as follows*

$$(v + t)(c) = v(c) + t, \tag{8a}$$

$$v[R](c) = \begin{cases} 0 & \text{for } c \in R, \\ v(c) & \text{otherwise.} \end{cases} \tag{8b}$$

We see that (8a) is used to progress time and that (8b) is used to reset the clocks in $R$ to zero.

Definition 5 (Semantics of Clock Constraint). *A clock constraint in $\Psi(C)$ is a set of clock valuations $\{v : C \to \mathbb{R}_{\geq 0}\}$ given by*

$$[\![c \bowtie k]\!] = \{v : C \to \mathbb{R}_{\geq 0} | v(c) \bowtie k\} \tag{9a}$$

$$[\![\psi_1 \wedge \psi_2]\!] = [\![\psi_1]\!] \cap [\![\psi_2]\!]. \tag{9b}$$

For convenience we denote $v \in [\![\psi]\!]$ by $v \models \psi$.

Definition 6 (Semantics of Timed Automaton). *The semantics of a timed automaton $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ is a transition system $[\![\mathcal{A}]\!] = (S, S_0, \Sigma \cup \mathbb{R}_{\geq 0}, T_\sigma \cup T_\mathrm{t})$, where*

$$\begin{aligned} S &= \{(e, v) \in E \times \mathbb{R}_{\geq 0}^C | v \models I(e)\} \\ S_0 &= \{(e, v) \in E_0 \times v_0\} \\ T_\sigma &= \{(e, v) \xrightarrow{\sigma} (e', v') | \exists (e, G_{e\to e'}, \sigma, R_{e\to e'}, e') \in \Delta : \\ &\qquad v \models G_{e\to e'},\ v' = v[R_{e\to e'}]\} \\ T_\mathrm{t} &= \{(e, v) \xrightarrow{t} (e, v + t) | \forall t' \in [0, t] : v + t' \models I(e)\}. \end{aligned}$$

Analog to the solution of (2), shown in (3), is a run of a timed automaton.

Definition 7 (Run of Timed Automaton). *A run of a timed automaton $\mathcal{A}$ is a possibly infinite sequence of alternations between time steps and discrete steps on the following form*

$$\varrho_\mathcal{A} : (e_0, v_0) \xrightarrow{t_1} (e_0, v_1) \xrightarrow{\sigma_1} (e_1, v_2) \xrightarrow{t_2} \dots \tag{10}$$

*which is a path in $[\![\mathcal{A}]\!]$, where $t_i \in \mathbb{R}_{\geq 0}$ and $\sigma_i \in \Sigma$. The multifunction describing the runs of a timed automaton $\phi_\mathcal{A} : \mathbb{R}_{\geq 0} \times E_0 \to 2^E$, is defined by $e \in \phi_\mathcal{A}(t, e_0)$ if and only if there exists a path in $[\![\mathcal{A}]\!]$ initialized in $(e_0, v_0)$ that reaches the location $e$ at time $t = \sum_i t_i$.*

From the run of a timed automaton, the reachable set is defined below.

Definition 8 (Reachable set of Timed Auto.). *The reachable set of a timed automaton $\mathcal{A}$, with initial locations $E_0$, in the time interval $[t_1, t_2]$ is defined as*

$$\begin{aligned} \mathrm{Reach}_{[t_1,t_2]}(\mathcal{A}, E_0) \equiv \{e \in E | \exists t \in [t_1, t_2], \exists e_0 \in E_0 \\ \text{such that } e \in \phi_\mathcal{A}(t, e_0)\}. \end{aligned} \tag{11}$$

A timed game automaton is closely related to a timed automaton as shown in the following [6].

Definition 9 (Timed Game Automaton). *A timed game automaton is a tuple $\mathcal{G} = (E, E_0, C, \Sigma_\mathrm{c}, \Sigma_\mathrm{u}, I, \Delta)$, where the tuple $(E, E_0, C, \Sigma_\mathrm{c} \cup \Sigma_\mathrm{u}, I, \Delta)$ is a timed automaton. The set $\Sigma_\mathrm{c}$ contains controllable actions (system inputs) and the set $\Sigma_\mathrm{u}$ contains uncontrollable actions (system outputs).*

Actions that can be affected by a strategy, see Definition 11, are controllable actions. However, transitions labeled with uncontrollable actions can happen whenever an adversary (the environment) chooses to take them and the associated guards are satisfied. In this setting, the input actions are equivalent to changing the control $g(x)$, and the output actions are observations, i.e., they resemble information about the state of the system.

Example 1. *Consider the timed game automaton shown in Figure 1 having four locations and one clock. It is initialized in location $e_1$ and the aim is to reach location $e_3$. Initially, an uncontrollable action can enable a transition to $e_2$, when the guard $c \geq 1$ is satisfied. After reaching $e_2$, the game is guaranteed to reach location $e_3$ if the controllable action $\sigma_\mathrm{c}$ happens when $c < 1$. Otherwise a transition to $e_4$ may occur.*

Analog to the controller $g(x)$, a strategy is defined in the following. Let us first define continuation of a run.

Definition 10 (Continuation of a Run). *Let $\varrho = (e_0, v_0) \to \dots \to (e_k, v_k)$ be a run. Then a continuation*

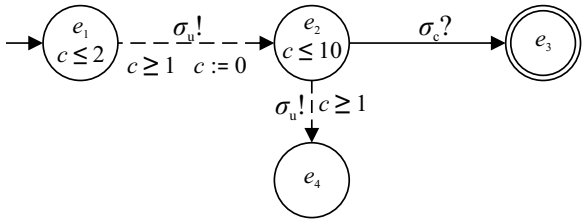


**Figure 1: Timed game automaton with four locations, where the solid line represents a transition with a controllable action and the dashed lines represent transitions with uncontrollable actions.**

*of the run $\varrho$ is the run $\varrho' = (e_0, v_0) \to \cdots \to (e_k, v_k) \to (e_{k+1}, v_{k+1})$, denoted $\varrho' = \varrho \to (e_{k+1}, v_{k+1})$.*

DEFINITION 11 (STRATEGY). *Let $\mathcal{G} = (E, E_0, C, \Sigma_c, \Sigma_u, I, \Delta)$ be a timed game automaton. Then any map $\kappa : S^+ \to \Sigma_c \cup \{\delta\}$, where $S^+$ is the set of all runs and $\delta \notin \Sigma_c \cup \Sigma_u$, which satisfies the following two conditions:*

1. *if $\kappa(\varrho) = \delta$, then $\varrho \xrightarrow{t} (e_k, v_k + t)$ is a path in $[\![\mathcal{G}]\!]$ for some $t > 0$, and*
2. *if $\kappa(\varrho) = \sigma$, then $\varrho \xrightarrow{\sigma} (e'_k, v'_k)$ is a path in $[\![\mathcal{G}]\!]$,*

*for any run $\varrho = (e_0, v_0) \to \cdots \to (e_k, v_k)$, is called a strategy.*

We see that the controller can either do nothing ($\kappa(\varrho) = \delta$) or execute a controllable action ($\kappa(\varrho) = \sigma$). Furthermore, we see that a strategy may depend on the entire past of the run. However, we are only interested in so-called memoryless strategies, i.e., strategies that only depend on the current state of the timed game automaton. This implies that if $\varrho = (e_0, v_0) \xrightarrow{t_1} (e_1, v_1) \to \cdots \to (e_k, v_k)$ and $\varrho' = (e_0, v_0) \xrightarrow{t'_1} (e'_1, v'_1) \to \cdots \to (e_k, v_k)$, then $\kappa(\varrho) = \kappa(\varrho')$. In addition to being memoryless, we require the strategy to be independent of the clock valuations, i.e., if $\varrho = (e_0, v_0) \xrightarrow{t_1} \cdots \to (e_k, v_k)$ and $\varrho' = (e_0, v_0) \xrightarrow{t'_1} (e'_1, v'_1) \to \cdots \to (e_k, v'_k)$, then $\kappa(\varrho) = \kappa(\varrho')$. A closed-loop system satisfying these properties can be implemented as a parallel composition of a timed automaton and an automaton.

DEFINITION 12. *Consider the timed game automaton $\mathcal{G}$ and the strategy $\kappa$, which only depends on the locations. Then $\mathcal{G}_\kappa$ is the timed automaton controlled using the strategy $\kappa$.*

# 3. GENERATION OF FINITE PARTITION

The presented abstraction is based on partitioning the state space and the control of $\Gamma = (X, U, f)$. The partitioning of the state space is inspired by the partitioning presented in [13]. However, in contrast to [13], the considered system has an unknown control of $u$, as stated in (1).

It is proposed to partition the state space by intersecting slices defined as the set-difference of positive and negative invariant sets. This implies that the partition should be conducted such that for each admissible control $g^i(x)$ the vector field of the controlled system $\Gamma_{g^i} = (X, f_{g^i})$ is transversal to the boundaries of the slices. The admissible controls are defined to be the finite set $K_U = \{g^i(x) | i \in \Lambda\}$, where $\Lambda$ is some index set. For simplicity, the controls in $K_U$ have domain $X$. However, this simplification can easily be relaxed.

DEFINITION 13 (SLICE). *A nonempty set $S$ is a slice if it is a union of cells and there exist two sets $A$ and $B$ that are positive or negative invariant such that*

1. *$B$ is a proper subset of $A$, i.e., $B \subset A$.*
2. *Given any $g \in K_U$, $A$ and $B$ are either positive or negative invariant sets for system $\Gamma_g = (X, f_g)$, and*
3. *$S = \mathrm{cl}(A \backslash B)$.*

From this definition, we see that for a given partition, only controls that make the vector field of the closed-loop system transversal to the boundaries of the slices are allowed.

To devise a partition of a state space, we need to define collections of slices, called slice-families.

DEFINITION 14 (SLICE-FAMILY). *A slice-family $\mathcal{S}$ is a collection of slices generated by the positive and negative invariant open sets $A_1 \subset A_2 \subset \cdots \subset A_k$ covering the entire state space of $\Gamma$, thereby $S_1 = A_1$, $S_2 = \mathrm{cl}(A_2 \backslash A_1), \ldots, S_k = \mathrm{cl}(A_k \backslash A_{k-1})$ and $X \subseteq A_k$. For convenience $|\mathcal{S}|$ is defined to be the cardinality of the slice-family $\mathcal{S}$, thus $\mathcal{S} = \{S_1, \ldots, S_{|\mathcal{S}|}\}$. Furthermore, we say that $\mathcal{S}$ is generated by the sets $\{A_i | i \in \boldsymbol{k}\}$.*

A function is associated to each slice-family $\mathcal{S}$, to provide an easy way of describing the boundary of a slice. Such functions are called partitioning functions.

DEFINITION 15 (PARTITIONING FUNCTION). *Let $\mathcal{S}$ be a slice-family, then a continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$ smooth on $\mathbb{R}^n \backslash \{0\}$ is a partitioning function associated to $\mathcal{S}$ if for any set $A_i$ generating $\mathcal{S}$ there exists $a_i, a'_i \in \mathbb{R} \cup \{-\infty, \infty\}$ such that*

$$\varphi^{-1}([a'_i, a_i]) = A_i \tag{12}$$

*and $a_i, a'_i$ are regular values of $\varphi$. By regular level set theorem, the boundary $\varphi^{-1}(a_i)$ of $A_i$ is a smooth manifold [15].*

In the remainder of the paper, we associate to each slice-family $\mathcal{S}^i$ a partitioning function $\varphi^i$.

The partition of the state space is associated with cells that are generated by intersecting slices.

DEFINITION 16. *We say that slices $S_1$ and $S_2$ intersect each other transversally and write*

$$S_1 \pitchfork S_2 = S_1 \cap S_2, \tag{13}$$

*if their boundaries, $\mathrm{bd}(S_1)$ and $\mathrm{bd}(S_2)$, intersect each other transversally.*

DEFINITION 17 (EXTENDED CELL). *Let* $\{\mathcal{S}^i | i \in \boldsymbol{k}\}$ *be a collection of* $k$ *slice-families and define* $\mathcal{Y} = \{1, \ldots, |\mathcal{S}^1|\} \times \cdots \times \{1, \ldots, |\mathcal{S}^k|\}$. *Denote the* $j^{\text{th}}$ *slice in* $\mathcal{S}^i$ *by* $S^i_j$ *and let* $y \in \mathcal{Y}$. *Then*

$$e_{\text{ex},y} \equiv \pitchfork_{i=1}^{k} S^i_{y_i}. \tag{14}$$

*Any nonempty set* $e_{\text{ex},y}$ *will be called an extended cell. These cells are denoted extended cells, since the transversal intersection of slices may form multiple disjoint sets.*

DEFINITION 18 (CELL). *A cell is a connected component of an extended cell*

$$\bigcup_z e_{(y,z)} = e_{\text{ex},y}, \text{ where} \tag{15a}$$

$$e_{(y,z)} \cap e_{(y,z')} = \emptyset \quad \forall z \neq z'. \tag{15b}$$

We say that the slices $S^1_{y_1}, \ldots, S^k_{y_k}$ generate the cell. In the remainder of the paper, we denote the slice from the $i^{\text{th}}$ slice-family generating $e$ by $S^i_e$.

PROPOSITION 1 (PROOF IN [14]). *If* $S_1 \pitchfork S_2 \neq \emptyset$ *then*

$$\text{int}(S_1 \cap S_2) \neq \emptyset. \tag{16}$$

A finite partition of the state space based on the transversal intersection of slices is defined in the following.

DEFINITION 19 (FINITE PARTITION OF STATE SPACE). *Let* $\mathcal{S}$ *be a collection of slice-families,* $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$. *Then the finite partition* $K_X(\mathcal{S})$ *is defined to be the collection of all cells generated by* $\mathcal{S}$ *according to Definition 18.*

Finally, the product of $K_X(\mathcal{S})$ and $K_U$ is defined to be the partition of $\Gamma$ given by $K(\mathcal{S}) \equiv K_X(\mathcal{S}) \times K_U$.

The following example clarifies how the partitioning is conducted.

EXAMPLE 2. *Consider the one-dimensional system with one control input and state space* $X = [-3, 3] \subset \mathbb{R}$

$$\dot{x} = -x + u. \tag{17}$$

*The partition of its state space is conducted using the sets* $A_1 = [-1, 1]$ *and* $A_2 = [-3, 3]$, *i.e., we obtain the cells:* $[-3, -1]$, $[-1, 1]$, $[1, 3]$. *The control of the system should also be partitioned, which is chosen to* $K_U = \{0, 1.5, 2x\}$. *Figure 2 shows a partition associated with* (17).

*From the figure it is seen that the direction of the vector field can be reversed completely or partly according to the input applied to the system. Furthermore, the two sets* $A_1 = [-1, 1]$ *and* $A_2 = [-3, 3]$ *are positive or negative invariant sets for all controls in* $K_U$.

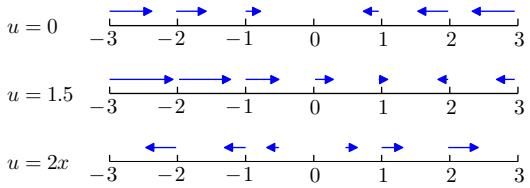


**Figure 2: Vector field of (17), illustrated with blue arrows, with three different controls applied to the system.**

# 4. GENERATION OF TIMED GAME FROM FINITE PARTITION

To abstract a control system $\Gamma$ by a timed game automaton $\mathcal{G}$, we modify the abstraction procedure presented in [13], by adding the distinction between controllable and uncontrollable actions according to Definition 9. Furthermore, we extend the expressiveness of the update maps of the timed game, to allow more accurate abstractions. First, an example is provided to illustrate the principle of the abstraction, then the abstraction procedure is presented.

The possibility to change the control input increases the number of locations and adds the possibility to influence the trajectories of the system, as shown in the following example.

EXAMPLE 3. *Consider a one-dimensional system from Example 2, where* $X = [-3, 3]$ *and*

$$\dot{x} = -x + u. \tag{18}$$

*The state space* $X$ *is partitioned into three cells, i.e.,* $K_X = \{[-3, -1], [-1, 1], [1, 3]\}$ *and the controls are in* $K_U = \{0, 1.5, 2x\}$. *All controls can be applied in all cells, i.e., the generated game has 9 locations* ($K_X \times K_U$). *A timed game abstracting this system is illustrated in Figure 3, where the execution of the controllable action* $\sigma^i_c$ *resembles applying the control* $u = g^i(x)$ *in the dynamical system.*

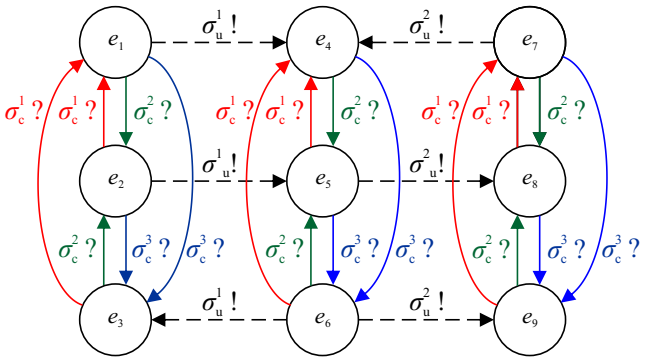


**Figure 3: Illustration of a timed game automaton, where three different controls can be applied to the system.**

Before providing the procedure for generating a timed game automaton, the clock valuation is redefined, as a more expressive update map is used in order to abstract the systems.

Furthermore, we define the clock valuation on pairs of clocks as the abstraction uses pairs of clocks to monitor the fastest respectively slowest progress in each direction.

DEFINITION 20 (EXTENDED CLOCK VALUATION). *Denote the pair of clocks $(c_1^i, c_2^i)$ by $c^i$, and let $C = \{c^i | i \in \boldsymbol{k}\}$. Then a clock valuation on a set of clocks $C$ is a mapping $v : C \to \mathbb{R}^2_{\geq 0}$. The initial valuation $v_0$ is given by $v_0(c) = (0,0)$ for all $c \in C$. For a valuation $v$, $t \in \mathbb{R}_{\geq 0}$, and $R \subseteq C$, the valuations $v + t$ and $v[R]$ on a pair of clocks $c$ are*

$$(v+t)(c) = v(c) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} t, \tag{19a}$$

$$v[R](c) = \alpha + \beta v(c) \text{ for } c \in R \tag{19b}$$

*where $\alpha \in \mathbb{R}^2$ and $\beta$ is a $2 \times 2$ matrix with rational entries.*

Note that resetting a pair of clocks is just a special case of (19b), where $\alpha = (0,0)$ and all entries of $\beta$ are zero. Furthermore, the valuation of one clock, denote $v(c_1^i)$ or $v(c_2^i)$, is also used in the following, when appropriate.

In the following, a procedure is presented for generating a timed game automaton from a finite partition.

PROCEDURE 1. *Given a partition $K(\mathcal{S})$, the timed game automaton $\mathcal{G} = (E, E_0, C, \Sigma_\mathrm{c}, \Sigma_\mathrm{u}, I, \Delta)$ is generated using the following*

- ***Locations:*** *Let the locations of $\mathcal{G}$ be given by*

$$E = K_X(\mathcal{S}) \times K_U. \tag{20}$$

*We use the notation $e_{(y,z,j)} \equiv (e_{(y,z)}, g^j) \in E$. This means that a location $e_{(y,z,j)}$ is associated with the cell $e_{(y,z)}$ of the partition $K_X(\mathcal{S})$ and the control $g^j(x)$ in $K_U$.*

- ***Clocks:*** *Given $k$ slice-families, the number of clocks is $2k$, i.e., $C = \{c^i | i \in \boldsymbol{k}\}$. The pair of clocks $c^i = (c_1^i, c_2^i)$ monitors the maximum and minimum time for being in slices of the slice-family $\mathcal{S}^i$.*

- ***Invariants:*** *In each location $e \in E$, there are up to $k$ invariants. The invariants for location $e_{(y,z,j)}$ specify upper bounds on the time for staying in the $k$ slices generating the cell $e_{(y,z)}$ with a control $g^j$ applied in $e_{(y,z)}$. We say that a cell $e_{(y,z)}$ is generated by the slices $\{S^i_{y_i} | i \in \boldsymbol{k}\}$ and in addition we say that a location $e_{(y,z,j)}$ is generated by $\{S^i_{(y_i,g_j)} | i \in \boldsymbol{k}\}$ and use the shorthand notation $S^i_e \equiv S^i_{(y_i,g_j)}$ for convenience. We impose an invariant whenever there is an upper bound on the time for staying in a slice generating the cell $e$*

$$I(e) = \bigwedge_{i=1}^{k} c_1^i \leq \overline{t}_{S_e^i} \tag{21}$$

*where $\overline{t}_{S_e^i} \in \mathbb{R}_{\geq 0}$ is an upper bound on the time for staying in $S_e^i$.*

- ***Controllable Actions:*** *The controllable actions $\Sigma_\mathrm{c}$ are actions $\sigma_\mathrm{c}^1, \ldots, \sigma_\mathrm{c}^{|K_U|}$, where $\sigma_\mathrm{c}^j$ is associated with applying the control $g^j(x)$ to the dynamical system.*

- ***Uncontrollable Actions:*** *The uncontrollable actions $\Sigma_\mathrm{u}$ are actions $\sigma_\mathrm{u}^1, \ldots, \sigma_\mathrm{u}^k$, where $\sigma_\mathrm{u}^i$ is associated with transitions between slices of the $i^{\text{th}}$ slice-family $\mathcal{S}^i = \{S_1^i, \ldots, S_{|\mathcal{S}^i|}^i\}$.*

- ***Transition relations:*** *If a pair of locations, $e$ and $e'$, where the control $g(x)$ is applied in both $e$ and $e'$, satisfy the following two conditions*

  1. *$e$ and $e'$ are adjacent cells in the state space, i.e., $e \cap e' \neq \emptyset$, with $S_e^i \neq S_{e'}^i$ for some $i \in \boldsymbol{k}$. Hence, $e$ and $e'$ are generated from different slices in $\mathcal{S}^i$, and*
  2. *$\varphi^i(x') \leq \varphi^i(x)\, \forall x \in e$ and $\forall x' \in e'$ and $\dot{\varphi}_g^i(x) < 0$ $\forall x \in e \cap e'$ or $\varphi^i(x') \geq \varphi^i(x)\, \forall x \in e$ and $\forall x' \in e'$ and $\dot{\varphi}_g^i(x) > 0$ $\forall x \in e \cap e'$.*

  *Then there is a transition relation*

$$\delta_{e \to e'} = (e, G_{e \to e'}, \sigma_\mathrm{u}^i, R_{\mathrm{u}, e \to e'}, e') \tag{22a}$$

  *where*

$$G_{e \to e'} = c_2^i \geq \underline{t}_{S_e^i} \tag{22b}$$

$$R_{\mathrm{u}, e \to e'} = c^i \tag{22c}$$

  *and $\underline{t}_{S_e^i} \in \mathbb{R}_{\geq 0}$ is a lower bound on the time for staying in $S_e^i$ and $v[R_{\mathrm{u},e\to e'}]$ is defined in (19b) with $\alpha = (0,0)$ and all entries of $\beta$ equal to zero. Note that $\sigma_\mathrm{u}^i$ is the action on the transition $\delta_{e\to e'}$, as $e$ and $e'$ are generated using different slices from the $i^{\text{th}}$ slice-family.*

  *At each location $e$, where the control $g^j(x)$ is applied, the following transitions are defined for all $i \in \{1, \ldots, |K_U|\} \backslash \{j\}$*

$$\delta_{e \to e'} = (e, G_{e \to e'}, \sigma_\mathrm{c}^i, R_{\mathrm{c}, e \to e'}, e'), \tag{23a}$$

  *where the control $g^i(x)$ is applied in $e'$ and*

$$G_{e \to e'} = \{c \geq 0 | c \in \{c_2^i | i \in \boldsymbol{k}\}\} \tag{23b}$$

$$R_{\mathrm{c}, e \to e'} = C. \tag{23c}$$

  *Note that there are no active guard conditions and that the exact values of $\alpha, \beta$ are provided in Theorem 1 in Section 5 to obtain soundness of the abstraction.*

For convenience the following notion is introduced.

DEFINITION 21. *Let $\mathcal{S}$ be a collection of slice-families, i.e., $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$. Then $\mathcal{G}(\mathcal{S})$ is the timed game automaton generated by $K(\mathcal{S})$ according to Procedure 1.*

REMARK 2. *Nonetheless, the locations of $\mathcal{G}(\mathcal{S})$ are associated with cells of $K_X(\mathcal{S})$, we will also utilize the timed game automaton $\mathcal{G}_{\mathrm{ex}}(\mathcal{S})$ with locations associated to extended cells, i.e., (recall the definition of $\mathcal{Y}$ from Definition 17)*

$$E = \{e_{\mathrm{ex},y} | y \in \mathcal{Y}\} \times K_U. \tag{24}$$

*The other steps of Procedure 1 are identical for the two timed game automata $\mathcal{G}(\mathcal{S})$ and $\mathcal{G}_{\mathrm{ex}}(\mathcal{S})$.*

## 5. PROPERTIES OF THE ABSTRACTION

The purpose of this section is to derive conditions for the partitions of the state space and control, and the conditions for the update maps under which an abstraction is sound.

To derive properties of the closed-loop system, we need a notion of controlled system, similar to $\mathcal{G}_\kappa$ from Definition 12.

DEFINITION 22. *Let the control system $\Gamma = (X, U, f)$ be controlled using the strategy $\kappa$ be denoted $\Gamma_\kappa$. Then the dynamics of $\Gamma_\kappa$ is given by*

$$\dot{x} = f(x, g(x)) \text{ where} \tag{25a}$$

$$g(x) = g^i(x) \, \forall x \in e_j \text{ iff } \kappa(e_j) = \sigma_{\mathrm{c}}^i. \tag{25b}$$

REMARK 3. *We assume that Filippov solutions do not occur, as the implementation of the timed automaton is based on integers in the guards and invariants. For more details about the problems that can occur when having Filippov solutions, see [11].*

A useful abstraction preserves safety. Therefore, the following is defined [2].

DEFINITION 23 (SOUND ABSTRACTION). *Let $\Gamma = (X, U, f)$ be a control system and suppose its state space $X$ is partitioned by $K_X(\mathcal{S}) = \{e_i | i \in \boldsymbol{k}\}$ and its control is partitioned by $K_U$. Let the initial states be $X_0 = \bigcup_{i \in \mathcal{I}} e_i$, with $\mathcal{I} \subseteq \boldsymbol{k}$. Then a timed game automaton $\mathcal{G} = (E = K_X \times K_U, E_0, C, \Sigma_{\mathrm{c}}, \Sigma_{\mathrm{u}}, I, \Delta)$ with $E = K_X(\mathcal{S}) \times K_U$ and $E_0 = \{e_i | i \in \mathcal{I}\} \times K_U$ is said to be a sound abstraction of $\Gamma$ on $[t_1, t_2]$ if $\forall t \in [t_1, t_2]$ and any strategy $\kappa$*

$$e_i \cap \mathrm{Reach}_{[t,t]}(\Gamma_\kappa, X_0) \neq \emptyset \text{ implies} \tag{26a}$$

$$\exists e_0 \in E_0 \text{ such that}$$

$$e_i \in \phi_{\mathcal{G}_\kappa}(t, e_0). \tag{26b}$$

*Note that the systems apply the same control at all time, as they follow the same strategy.*

If a sound abstraction $\mathcal{G}$ is safe for some strategy $\kappa$ then $\Gamma$ is also safe for the same strategy, as $\mathcal{G}_\kappa$ reaches all locations reached by $\Gamma_\kappa$.

DEFINITION 24 (COMPLETE ABSTRACTION). *Let $\Gamma$ be a control system and suppose its state space $X$ is partitioned by $K_X(\mathcal{S}) = \{e_i | i \in \boldsymbol{k}\}$ and its control is partitioned by $K_U$ and let the initial states be $X_0 = \bigcup_{i \in \mathcal{I}} e_i$, with $\mathcal{I} \subseteq \boldsymbol{k}$. Then a timed game automaton $\mathcal{G} = (E, E_0, C, \Sigma_{\mathrm{c}}, \Sigma_{\mathrm{u}}, I, \Delta)$ with $E = K_X(\mathcal{S}) \times K_U$ and $E_0 = \{e_i | i \in \mathcal{I}\} \times K_U$ is said to be a complete abstraction of $\Gamma$ on $[t_1, t_2]$ if it is a sound abstraction and $\forall t \in [t_1, t_2]$, any $\kappa$, and*

$$\text{for each } e_i \in \mathrm{Reach}_{[t,t]}(\mathcal{G}_\kappa, E_0) \tag{27a}$$

$$\exists x_0 \in X_0 \text{ such that}$$

$$\phi_{\Gamma_\kappa}(t, x_0) \in e_i. \tag{27b}$$

A complete abstraction $\mathcal{G}_\kappa$ is safe (unsafe) if and only if $\Gamma_\kappa$ is also safe (unsafe).

This next proposition follows directly from Proposition 6 in [13].

PROPOSITION 2. *A timed game automaton $\mathcal{G}_{\mathrm{ex}}(\mathcal{S}) = \mathcal{G}_1(\mathcal{S}^1) || \ldots || \mathcal{G}_k(\mathcal{S}^k)$, with locations abstracting extended cells, is a sound (complete) abstraction of the control system $\Gamma$ if and only if $\mathcal{G}_1(\mathcal{S}^1), \ldots, \mathcal{G}_k(\mathcal{S}^k)$ are sound (complete) abstractions of $\Gamma$.*

Let $\mathcal{S}$ be a slice family. We say that a control $g$ is an admissible control if for each slice $S \in \mathcal{S}$ we have either $\dot{\varphi}_g(x) > 0$ for all $x \in S \backslash \mathrm{Cr}(f_g)$ or $\dot{\varphi}_g(x) < 0$ for all $x \in S \backslash \mathrm{Cr}(f_g)$. We introduce the notation $\dot{\varphi}_g > 0$ on $S$ if and only if $\dot{\varphi}_g(x) > 0$ for some, thus, for all $x \in S \backslash \mathrm{Cr}(f_g)$.

LEMMA 1. *Let $\mathcal{S}$ be a slice family on $\mathbb{R}^n$, and $\varphi$ be a partitioning function associated to $\mathcal{S}$. Let $\{t_j \in \mathbb{R}_{\geq 0} | \ j \in \boldsymbol{k} \cup \{0\}\}$ be a sequence of nondecreasing real numbers, and $\{g_j : \mathbb{R}^m \to \mathbb{R}^n | \ j \in \boldsymbol{k}\}$ be a sequence of controls, where $g_j$ is applied for $t \in [t_{j-1}, t_j]$.*

*Suppose $S \in \mathcal{S}$. For convenience, let $\underline{\dot{\varphi}}_{g_j} \equiv \inf\{|\dot{\varphi}_{g_j}(x)| \in \mathbb{R}_{\geq 0} | x \in S\}$, $\overline{\dot{\varphi}}_{g_j} \equiv \sup\{|\dot{\varphi}_{g_j}(x)| \in \mathbb{R}_{\geq 0} | x \in S\}$, $\Delta_k(t) \equiv t - t_{k-1}$, $J^+ \equiv \{j \in \boldsymbol{k} | \dot{\varphi}_{g_j} > 0\}$, and $J^- \equiv \boldsymbol{k} \backslash J^+$.*

*If for all $j \in \boldsymbol{k}$*

$$\mathrm{Reach}_{[t_{j-1}, t_j]}(\Gamma_{g_j}, x_{j-1}) \subset S \tag{28}$$

*then for all $t \in [t_{k-1}, t_k]$*

$$\begin{aligned}
&\sum_{j \in J^+ \backslash \{k\}} \Delta_j(t_j) \underline{\dot{\varphi}}_{g_j} - \sum_{j \in J^- \backslash \{k\}} \Delta_j(t_j) \overline{\dot{\varphi}}_{g_j} \\
&+ \begin{cases} \Delta_k(t) \underline{\dot{\varphi}}_{g_k} \text{ if } \dot{\varphi}_{g_k} > 0 \\ -\Delta_k(t) \overline{\dot{\varphi}}_{g_k} \text{ if } \dot{\varphi}_{g_k} < 0 \end{cases} \\
&\leq \sum_{j=1}^{k-1} \int_{t_{j-1}}^{t_j} \dot{\varphi}_{g_j}(\phi_{\Gamma_{g_j}}(\Delta_j(\tau), x_{j-1})) d\tau \\
&+ \int_{t_{k-1}}^{t} \dot{\varphi}_{g_k}(\phi_{\Gamma_{g_k}}(\Delta_k(\tau), x_{k-1})) d\tau \\
&\leq \sum_{j \in J^+ \backslash \{k\}} \Delta_j(t_j) \overline{\dot{\varphi}}_{g_j} - \sum_{j \in J^- \backslash \{k\}} \Delta_j(t_j) \underline{\dot{\varphi}}_{g_j} \\
&+ \begin{cases} \Delta_k(t) \overline{\dot{\varphi}}_{g_k} \text{ if } \dot{\varphi}_{g_k} > 0 \\ -\Delta_k(t) \underline{\dot{\varphi}}_{g_k} \text{ if } \dot{\varphi}_{g_k} < 0 \end{cases} .
\end{aligned} \tag{29}$$

*Note that $x_j \equiv \phi_{\Gamma_{g_j}}(\Delta_j(t_j), x_{j-1})$.*

PROOF. The inequalities in (29) are the consequence of $\underline{\dot{\varphi}}_{g_j} \leq \dot{\varphi}_{g_j}(x)$ and $\overline{\dot{\varphi}}_{g_j} \geq \dot{\varphi}_{g_j}(x)$ for all $x \in S$. □

The principle of the lemma is illustrated in Figure 4, where $\varphi(\phi_\Gamma(\Delta_j(\tau), x_{j-1}))$ for $j = 1, \ldots, 4$ (black line) is plotted together with the upper approximation (red) and lower approximation (blue). It is seen that the inaccuracy of the approximation increases with time.

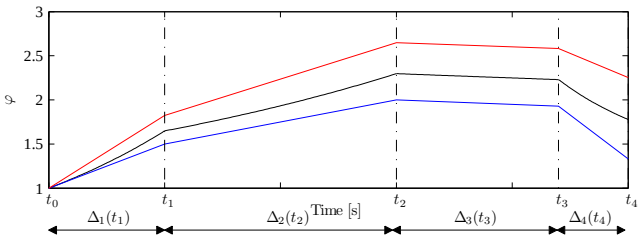


**Figure 4: The value of the Lyapunov function evaluated along a solution curve (black) and an upper approximation (red) and a lower approximation (blue) of it.**

We use Lemma 1 to set up invariants and guards for the solution to stay in a slice $S$. Suppose $\varphi^{-1}([1,3])$ in Figure 4 is a slice, then Corollary determines the time, when the red and blue lines intersect $\varphi^{-1}(1)$ and $\varphi^{-1}(3)$.

COROLLARY 1. *Let $\mathcal{S}$ be a slice family on $\mathbb{R}^n$, and $\varphi$ be a partitioning function associated to $\mathcal{S}$. Suppose $S \in \mathcal{S}$ and $S = \varphi^{-1}([a_{h-1}, a_h])$, and define $\Delta a \equiv a_h - a_{h-1}$. Furthermore, assume that $x_0 \in \varphi^{-1}(a_{h-1})$ and that $\dot{\varphi}_{g_1} > 0$. Then from (29) it follows that for all $t \in [t_{k-1}, t_k]$ with $k \in J^+$, an invariant for the solution to stay in the slice $S$ is*

$$\sum_{j\in J^+\setminus\{k\}} \Delta_j(t_j)\underline{\dot{\varphi}}_{g_j} - \sum_{j\in J^-\setminus\{k\}} \Delta_j(t_j)\overline{\dot{\varphi}}_{g_j} + \Delta_k(t)\underline{\dot{\varphi}}_{g_k} \leq \Delta a. \tag{30}$$

*This implies that if inequality (30) is violated, then $\phi_{\Gamma_{g_k}}(t - t_{k-1}, x_{k-1}) \notin S$. Similarly, for all $t \in [t_{k-1}, t_k]$, with $k \in J^+$ a guard for the solution to stay in the slice is*

$$\sum_{j\in J^+\setminus\{k\}} \Delta_j(t_j)\overline{\dot{\varphi}}_{g_j} - \sum_{j\in J^-\setminus\{k\}} \Delta_j(t_j)\underline{\dot{\varphi}}_{g_j} + \Delta_k(t)\overline{\dot{\varphi}}_{g_k} \geq \Delta a. \tag{31}$$

*This implies that if (31) is violated, then $\phi_{\Gamma_{g_k}}(t - t_{k-1}, x_{k-1}) \in S$.*

*Additionally, for all $t \in [t_{k-1}, t_k]$, with $k \in J^-$ an invariant for the solution to stay in the slice is*

$$\sum_{j\in J^+\setminus\{k\}} \Delta_j(t_j)\overline{\dot{\varphi}}_{g_j} - \sum_{j\in J^-\setminus\{k\}} \Delta_j(t_j)\underline{\dot{\varphi}}_{g_j} - \Delta_k(t)\underline{\dot{\varphi}}_{g_k} \geq 0. \tag{32}$$

*Finally, for all $t \in [t_{k-1}, t_k]$, with $k \in J^-$ a guard for the solution to stay in the slice is*

$$\sum_{j\in J^+\setminus\{k\}} \Delta_j(t_j)\underline{\dot{\varphi}}_{g_j} - \sum_{j\in J^-\setminus\{k\}} \Delta_j(t_j)\overline{\dot{\varphi}}_{g_j} - \Delta_k(t)\overline{\dot{\varphi}}_{g_k} \leq 0. \tag{33}$$

PROOF. Note that if $\dot{\varphi}_{g_k} > 0$, the solution leaves the slice at some $t \in [t_{k-1}, t_k]$ when $\phi_{\Gamma_{g_j}}(t - t_{k-1}, x_{k-1}) \in \varphi^{-1}(a_h)$, i.e., for some $t \in [t_{k-1}, t_k]$

$$\sum_{j=1}^{k-1} \int_{t_{j-1}}^{t_j} \dot{\varphi}_{g_j}(\phi_{\Gamma_{g_j}}(\Delta_j(\tau), x_{j-1}))d\tau + \int_{t_{k-1}}^{t} \dot{\varphi}_{g_k}(\phi_{\Gamma_{g_k}}(\Delta_k(\tau), x_{k-1}))d\tau = \Delta a. \tag{34}$$

If $\dot{\varphi}_{g_k} < 0$, the solution leaves the slice when $\phi_{\Gamma_{g_k}}(\Delta_k(t), x_{k-1}) \in \varphi^{-1}(a_{h-1})$ for some $t \in [t_{k-1}, t_k]$, i.e., for some $t \in [t_{k-1}, t_k]$

$$\sum_{j=1}^{k-1} \int_{t_{j-1}}^{t_j} \dot{\varphi}_{g_j}(\phi_{\Gamma_{g_j}}(\Delta_j(\tau), x_{j-1}))d\tau + \int_{t_{k-1}}^{t} \dot{\varphi}_{g_k}(\phi_{\Gamma_{g_k}}(\Delta_k(\tau), x_{k-1}))d\tau = 0. \tag{35}$$

This provides the right hand sides of (30)-(33). However, note that $\leq$ ($\geq$) also changes to $\geq$ ($\leq$), which is due to the changing direction of $\dot{\varphi}_{g_j}$. □

The corollary provides guard and invariant conditions, which give the minimum and maximum time a trajectory stays within a slice for a given sequence of controls.

A sufficient and necessary condition for soundness of an abstraction is formulated in the following. To stress that the control is of importance, we denote $S^i_e$, used in (21) and (22b), by $S^i_{(y_i,g)}$.

THEOREM 1. *A timed game automaton $\mathcal{G}(\mathcal{S}) = (E, E_0, C, \Sigma_c, \Sigma_u, I, \Delta)$ is a sound abstraction of the control system $\Gamma$, if and only if its invariants and guards are given by (21) and (22b), where for each $y \in \mathcal{Y}$ and $g \in K_U$*

$$\underline{t}_{S^i_{(y_i,g)}} \leq \frac{|a^i_{y_i} - a^i_{y_i-1}|}{\sup\{|\dot{\varphi}^i_g(x)| \in \mathbb{R}_{\geq 0} | x \in S^i_{y_i}\}} \tag{36a}$$

$$\overline{t}_{S^i_{(y_i,g)}} \geq \frac{|a^i_{y_i} - a^i_{y_i-1}|}{\inf\{|\dot{\varphi}^i_g(x)| \in \mathbb{R}_{\geq 0} | x \in S^i_{y_i}\}} \tag{36b}$$

*and $\dot{\varphi}^i_g(x)$ is defined as shown in (4a). The update map for transitions relations associated with controllable actions, see (23c), between two locations $e$ and $e'$ with control $g$ respectively $g'$ is*

$$v[R_{c,e\to e'}] \equiv \begin{cases} \begin{bmatrix} \frac{\overline{t}_{S^i_{(y_i,g')}}}{\overline{t}_{S^i_{(y_i,g)}}} & 0 \\ 0 & \frac{\underline{t}_{S^i_{(y_i,g')}}}{\underline{t}_{S^i_{(y_i,g)}}} \end{bmatrix} v(c^i) \text{ if } \dot{\varphi}_g\dot{\varphi}_{g'} > 0 \\ \begin{bmatrix} \overline{t}_{S^i_{(y_i,g')}} \\ \underline{t}_{S^i_{(y_i,g')}} \end{bmatrix} - \begin{bmatrix} 0 & \frac{\overline{t}_{S^i_{(y_i,g')}}}{\underline{t}_{S^i_{(y_i,g)}}} \\ \frac{\underline{t}_{S^i_{(y_i,g')}}}{\overline{t}_{S^i_{(y_i,g)}}} & 0 \end{bmatrix} v(c^i) \text{ otherwise.} \end{cases} \tag{37}$$

PROOF. In this proof, we show by induction that the invariants and guards imposed on the clocks of $\mathcal{G}(\mathcal{S})$ generated

using Theorem 1, where

$$\underline{t}_{S^i_{(y_i,g)}} = \frac{|a^i_{y_i} - a^i_{y_i-1}|}{\sup\{|\dot{\varphi}^i_g(x)| \in \mathbb{R}_{\geq 0} | x \in S^i_{y_i}\}} \tag{38a}$$

$$\overline{t}_{S^i_{(y_i,g)}} = \frac{|a^i_{y_i} - a^i_{y_i-1}|}{\inf\{|\dot{\varphi}^i_g(x)| \in \mathbb{R}_{\geq 0} | x \in S^i_{y_i}\}} \tag{38b}$$

are equivalent to the guards and invariants given by Corollary 1. As Corollary 1 gives conditions for a sound approximation this will prove the theorem.

Recall that $S^i_{y_i} = (\varphi^i)^{-1}([a^i_{y_i-1}, a^i_{y_i}])$, where $a^i_{y_i-1} < a^i_{y_i}$ and $\Delta a = a^i_{y_i} - a^i_{y_i-1}$. Assume that $x_0 \in (\varphi^i)^{-1}(a^i_{y_i-1})$, $\dot{\varphi}^i_{g_1} > 0$, and $v_0(c^i) = (0,0)$. Note that the valuation of the clocks are assumed to be zero, as $x_0$ is on the boundary of the considered slice.

First, we show that the invariants of Corollary 1 and Theorem 1 are the same for all $t \in [t_0, t_1]$; secondly, we assume they are the same for all $t \in [t_{k-1}, t_k]$. Finally, we show that they are the same for all $t \in [t_k, t_{k+1}]$.

Base case: We show that for $t \in [t_0, t_1]$ the guards and invariants in (30)-(33), shown in (39) for the considered case, are equivalent to the guard and invariant generated by using Theorem 1. From (30)-(33) we know that for all $t \in [t_0, t_1]$

$$(t - t_0)\underline{\dot{\varphi}}_{g_1} \leq \Delta a \Leftrightarrow (t - t_0) \leq \overline{t}_{g_1} \tag{39a}$$

$$(t - t_0)\overline{\dot{\varphi}}_{g_1} \geq \Delta a \Leftrightarrow (t - t_0) \geq \underline{t}_{g_1}. \tag{39b}$$

The invariants and guards of the abstraction are $c^i_1 \leq \overline{t}_{g_1}$ and $c^i_2 \geq \underline{t}_{g_1}$ where the clock valuations for all $t \in [t_0, t_1]$ are

$$v(c^i_1) \leq \overline{t}_{g_1} \Leftrightarrow v_0(c^i_1) + (t - t_0) \leq \overline{t}_{g_1} \tag{40a}$$

$$v(c^i_2) \geq \underline{t}_{g_1} \Leftrightarrow v_0(c^i_2) + (t - t_0) \geq \underline{t}_{g_1}. \tag{40b}$$

It is seen that (39) and (40) are equivalent, as $v_0(c^i) = (0,0)$.

Inductive step: Assume that the guards and invariants of Theorem 1 are equivalent to the guards and invariants derived from Corollary 1 for all $t \in [t_{k-1}, t_k]$ and that $k \in J^+$

$$v(c^i_1) \leq \overline{t}_{g_k} \Leftrightarrow \sum_{j \in J^+ \setminus \{k\}} \Delta_j(t_j)\underline{\dot{\varphi}}_{g_j} - \sum_{j \in J^- \setminus \{k\}} \Delta_j(t_j)\overline{\dot{\varphi}}_{g_j} + \Delta_k(t)\underline{\dot{\varphi}}_{g_k} \leq \Delta a \tag{41a}$$

$$v(c^i_2) \geq \underline{t}_{g_k} \Leftrightarrow \sum_{j \in J^+ \setminus \{k\}} \Delta_j(t_j)\overline{\dot{\varphi}}_{g_j} - \sum_{j \in J^- \setminus \{k\}} \Delta_j(t_j)\underline{\dot{\varphi}}_{g_j} + \Delta_k(t)\overline{\dot{\varphi}}_{g_k} \geq \Delta a \tag{41b}$$

and if $k \in J^-$, then

$$v(c^i_1) \leq \overline{t}_{g_k} \Leftrightarrow \sum_{j \in J^+ \setminus \{k\}} \Delta_j(t_j)\overline{\dot{\varphi}}_{g_j} - \sum_{j \in J^- \setminus \{k\}} \Delta_j(t_j)\underline{\dot{\varphi}}_{g_j} - \Delta_k(t)\underline{\dot{\varphi}}_{g_k} \geq 0 \tag{42a}$$

$$v(c^i_2) \geq \underline{t}_{g_k} \Leftrightarrow \sum_{j \in J^+ \setminus \{k\}} \Delta_j(t_j)\underline{\dot{\varphi}}_{g_j} - \sum_{j \in J^- \setminus \{k\}} \Delta_j(t_j)\overline{\dot{\varphi}}_{g_j} - \Delta_k(t)\overline{\dot{\varphi}}_{g_k} \leq 0 \tag{42b}$$

Now we show that the guards and invariants imposed by Theorem 1 are still equivalent to those of Corollary 1 at $j = k+1$. To shorten the proof, we only show this in two of the possible cases ($\dot{\varphi}_{g_k}\dot{\varphi}_{g_{k+1}} > 0$ and $\dot{\varphi}_{g_k} > 0$, $\dot{\varphi}_{g_k}\dot{\varphi}_{g_{k+1}} > 0$ and $\dot{\varphi}_{g_k} < 0$, $\dot{\varphi}_{g_k}\dot{\varphi}_{g_{k+1}} < 0$ and $\dot{\varphi}_{g_k} > 0$, $\dot{\varphi}_{g_k}\dot{\varphi}_{g_{k+1}} < 0$ and $\dot{\varphi}_{g_k} < 0$).

Assume that $\dot{\varphi}_{g_k}\dot{\varphi}_{g_{k+1}} > 0$ and $\dot{\varphi}_{g_k} > 0$. Then the invariant assumed in Theorem 1 becomes for all $t \in [t_k, t_{k+1}]$

$$\frac{\overline{t}_{g_{k+1}}}{\overline{t}_{g_k}} v_k(c^i_1) + \Delta_k(t) \leq \overline{t}_{g_{k+1}} \tag{43a}$$

$$\frac{\underline{\dot{\varphi}}_{g_k}}{\underline{\dot{\varphi}}_{g_{k+1}}} v_k(c^i_1) + \Delta_k(t) \leq \overline{t}_{g_{k+1}} \tag{43b}$$

We use (41a) to obtain $v_k(c^i_1)$ by dividing the inequality by $\underline{\dot{\varphi}}_{g_k}$

$$v_k(c^i_1) = \sum_{j \in J^+} \Delta_j(t_j)\frac{\underline{\dot{\varphi}}_{g_j}}{\underline{\dot{\varphi}}_{g_k}} - \sum_{j \in J^-} \Delta_j(t_j)\frac{\overline{\dot{\varphi}}_{g_j}}{\underline{\dot{\varphi}}_{g_k}} \tag{44}$$

Inserting (44) into (43b) and multiplying it by $\underline{\dot{\varphi}}_{g_{k+1}}$ yields for all $t \in [t_k, t_{k+1}]$ (Note that $\Delta a = \overline{t}_{g_{k+1}}\underline{\dot{\varphi}}_{g_{k+1}}$)

$$\sum_{j \in J^+} \Delta t_j \underline{\dot{\varphi}}_{g_j} - \sum_{j \in J^-} \Delta t_j \overline{\dot{\varphi}}_{g_j} + \underline{\dot{\varphi}}_{g_{k+1}} \Delta_k(t) \leq \Delta a. \tag{45}$$

This equals (41a) for $j = k+1$.

Now assume that $\dot{\varphi}_{g_k}\dot{\varphi}_{g_{k+1}} < 0$ and $\dot{\varphi}_{g_k} > 0$. The guard generated by the assumption of Theorem 1 becomes for all $t \in [t_k, t_{k+1}]$

$$\underline{t}_{g_{k+1}} - \frac{\underline{t}_{g_{k+1}}}{\overline{t}_{g_k}} v_k(c^i_1) + \Delta_k(t) \geq \underline{t}_{g_{k+1}} \tag{46a}$$

$$\frac{\Delta a}{\overline{\dot{\varphi}}_{g_{k+1}}} - \frac{\underline{\dot{\varphi}}_{g_k}}{\overline{\dot{\varphi}}_{g_{k+1}}} v_k(c^i_1) + \Delta_k(t) \geq \underline{t}_{g_{k+1}}. \tag{46b}$$

We use (41a) to obtain $v_k(c^i_1)$ by dividing the inequality by $\underline{\dot{\varphi}}_{g_k}$

$$v_k(c^i_1) = \sum_{j \in J^+} \Delta_j(t_j)\frac{\underline{\dot{\varphi}}_{g_j}}{\underline{\dot{\varphi}}_{g_k}} - \sum_{j \in J^-} \Delta_j(t_j)\frac{\overline{\dot{\varphi}}_{g_j}}{\underline{\dot{\varphi}}_{g_k}}. \tag{47}$$

Inserting (47) into (46b) and multiplying it with $\overline{\dot{\varphi}}_{g_{k+1}}$ yields

for all $t \in [t_k, t_{k+1}]$ (Note that $\Delta a = \underline{t}_{g_{k+1}} \overline{\varphi}_{g_{k+1}}$)

$$\Delta a - \sum_{j \in J^+} \Delta_j(t_j) \underline{\dot{\varphi}}_{g_j} + \sum_{j \in J^-} \Delta_j(t_j) \overline{\dot{\varphi}}_{g_j} + \overline{\dot{\varphi}}_{g_{k+1}} \Delta_k(t) \geq \Delta a \tag{48a}$$

$$\sum_{j \in J^+} \Delta_j(t_j) \underline{\dot{\varphi}}_{g_j} - \sum_{j \in J^-} \Delta_j(t_j) \overline{\dot{\varphi}}_{g_j} - \overline{\dot{\varphi}}_{g_{k+1}} \Delta_k(t) \leq 0 \tag{48b}$$

This equals (42b) for $j = k + 1$. □

REMARK 4. *The theorem provides the closest possible sound abstraction using only functions with constant derivatives, when* (38) *is satisfied. This follows from Figure 4, where it is seen that the lower and upper approximations can actually be a tangent to the Lyapunov function in the beginning of a time interval. Hence, if the conditions are strengthened, the abstraction will no longer be a sound approximation of the control system.*

## 6. ILLUSTRATIVE EXAMPLE

In this section, we apply the abstraction on a model of a unicycle given as

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} u \tag{49}$$

where $x \in \mathbb{R}^4$ and $u \in \mathbb{R}^2$. The first coordinate $x_1$ is the position of the unicycle in the $x$-direction, $x_2$ is the velocity of the unicycle in the $x$-direction, $x_3$ is the position of the unicycle in the $y$-direction, and $x_4$ is the velocity of the unicycle in the $y$-direction. The inputs are the acceleration in the $x$-direction ($u_1$) and the acceleration in the $y$-direction ($u_2$). The dynamics of this system is not complex contrary to the control objective is. The objective is to design a controller, ensuring that the system always reaches the goal set (green) from the initial set (blue), without hitting any of the obstacles (red), see Figure 5.

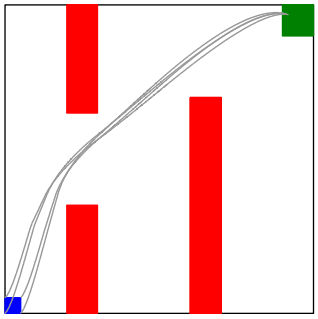

**Figure 5: Track of the unicycle, given by initial set (blue), goal set (green), and obstacles (red). The gray lines are trajectories of the robot, controlled using the proposed strategy.**

The partition is conducted using four Lyapunov functions, which are not shown, as they are four-dimensional.

A control law, given by a switched controller, is generated from the abstraction, where the switching is determined by the cell that the system is in. From this example, it is concluded that it is possible to generate control strategies from the proposed abstraction. However, it is not automatically generated, as the update maps introduced in this paper are not implemented in currently available verification tools.

## 7. CONCLUSION

In this paper, a method for abstracting control systems by timed games has been presented. The method is based on partitioning the state space of the systems by set-differences of sets that are positive or negative invariant for all admissible controls. The timed games used in the abstraction have a more expressive update map than the update map usually allowed for timed game automata. This makes it possible to generate the proposed sound approximations, but has the consequence that no tools exists for automatic controller synthesis.

To enable synthesis of a control strategy that ensures safety of the system, conditions for soundness have been set up. These conditions tell when a sound abstraction can be realized from a partition of a state space and a partition of the control. Finally, an example is provided to demonstrate that the formalism can be used to synthesize controllers that satisfy temporal specifications.

## 8. REFERENCES


[1] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Logic in Computer Science, 1990. LICS '90, Proceedings., Fifth Annual IEEE Symposium on*, pages 414–425, jun. 1990.

[2] R. Alur, T. Dang, and F. Ivancic. Progress on reachability analysis of hybrid systems using predicate abstraction. In *Proceedings of Hybrid Systems: Computation and Control*, pages 35–48, 2003.

[3] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symp. System Structure and Control*, pages 469–474, 1998.

[4] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR 05, LNCS 3653*, pages 66–80. Springer, 2005.

[5] F. Clarke, Y. Ledyaev, R. Stern, and P. Wolenski. *Nonsmooth Analysis and Control Theory*. Springer, 1998.

[6] A. David, K. G. Larsen, S. Li, and B. Nielsen. A game-theoretic approach to real-time system testing. In *DATE*, pages 486–491, Munich, Germany, March 2008.

[7] U. Fahrenberg, K. G. Larsen, and C. R. Thrane. Verification, performance analysis and controller synthesis for real-time systems. In *FSEN*, pages 34–61, 2009.

[8] G. E. Fainekos, A. Girard, and G. J. Pappas. Hierarchical synthesis of hybrid controllers from temporal logic specifications. In *Hybrid Systems: Computation and Control*, pages 203–216. Springer, 2007.

[9] A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *Automatic Control, IEEE*

*Transactions on*, 55(1):116–126, jan. 2010.

[10] L. Habets and J. van Schuppen. Control to facet problems for affine systems on simplices and polytopes - with applications to control of hybrid systems. In *CDC-ECC '05*, pages 4175–4180, dec. 2005.

[11] J. Leth and R. Wisniewski. On formalism and stability of switched systems. *Submitted to Journal of Control Theory and Applications*.

[12] K. R. Meyer. Energy functions for morse smale systems. *American Journal of Mathematics*, 90(4):1031–1040, 1968.

[13] C. Sloth and R. Wisniewski. Abstraction of continuous dynamical systems utilizing lyapunov functions. In *Proceedings of the 49th IEEE Conference on Decision and Control*, pages 3760–3765, Atlanta, Georgia, USA, December 2010.

[14] C. Sloth and R. Wisniewski. Proofs for an abstraction of continuous dynamical systems utilizing lyapunov functions. *arXiv:1008.3222*, 2010.

[15] L. W. Tu. *An Introduction to Manifolds*. Springer, 2008.